Diffie–Hellman key exchange

*Introduction :-* Hello Today our team is going to show our implementation of Diffie–Hellman key exchange algorithm.

Team Members :-

Kedar Shidhaye 16010121187

Anirudha Ta 16010121199

Aditya Ved 16010121208

The diffie hellman key exchange algorithm is used to securely exchange cryptographic keys in an insecure public channel previously for any kind of secure encrypted communication keys needed to be exchanged through some sort of physical means like courier service, in person meetings etc. Let's say we have 2 parties who have no prior knowledge of each other. In this scenario Diffie hellman helps by jointly establishing a shared secret key over an insecure channel from the secret private keys of the 2 parties and the private keys don't need to be exposed on the public network in this process.

*Features :-*

1. Key establishment without prior secrets: Unlike traditional methods, Diffie Hellman doesn't require pre-shared secrets, making it ideal for initial communication channels.
2. Public key exchange: Participants publicly exchange calculated values based on their private keys, but the actual private keys remain secret.
3. Shared secret generation: Both parties derive the same shared secret key from the exchanged values, even though they use different private keys.
4. Ephemeral keys: Often used with short-lived, temporary private keys for added security.(Ephemeral keys - short-lived, temporary private keys generated and used only for a single key exchange session.)

*Characteristics :-*

1. Susceptible to man-in-the-middle attacks: Requires additional mechanisms like digital signatures to ensure authenticity and prevent eavesdropping.
2. Standardized and widely used: Used in various protocols like TLS, SSH, and IKEv2 for secure communication channels.
3. Mathematical foundation: The algorithm relies on the difficulty of computing discrete logarithms in finite fields, making it computationally difficult to decipher and provides robust security.
4. Provides Strong Keys for Advanced Encryption Techniques: Deffie Hellman can use primes up to length of 2048 bits. As a result the generated key can be used for Advanced Encryption Techniques like AES or DES to securely encrypt and share messages over insecure channels

*Methodology :-*

In diffie hellman key exchange algorithm the necessary components are 2 parties with their secret private keys and 2 public keys (g,p) that the parties need to agree upon. Here p is a large 2048-3072 bit prime number and g is the primitive root of p.  
Note : **primitive root** - A primitive root of a prime number p is a number g such that all the numbers 1,2,..., p−1 can be written as for some value of n.

To determine how to choose the p & g as we can’t just randomly generate a large prime as primes basically become rarer and rarer as the number grows larger and larger we want a 2048 bit prime which runs in the range.([prime number theorem](https://en.wikipedia.org/wiki/Prime_number_theorem))

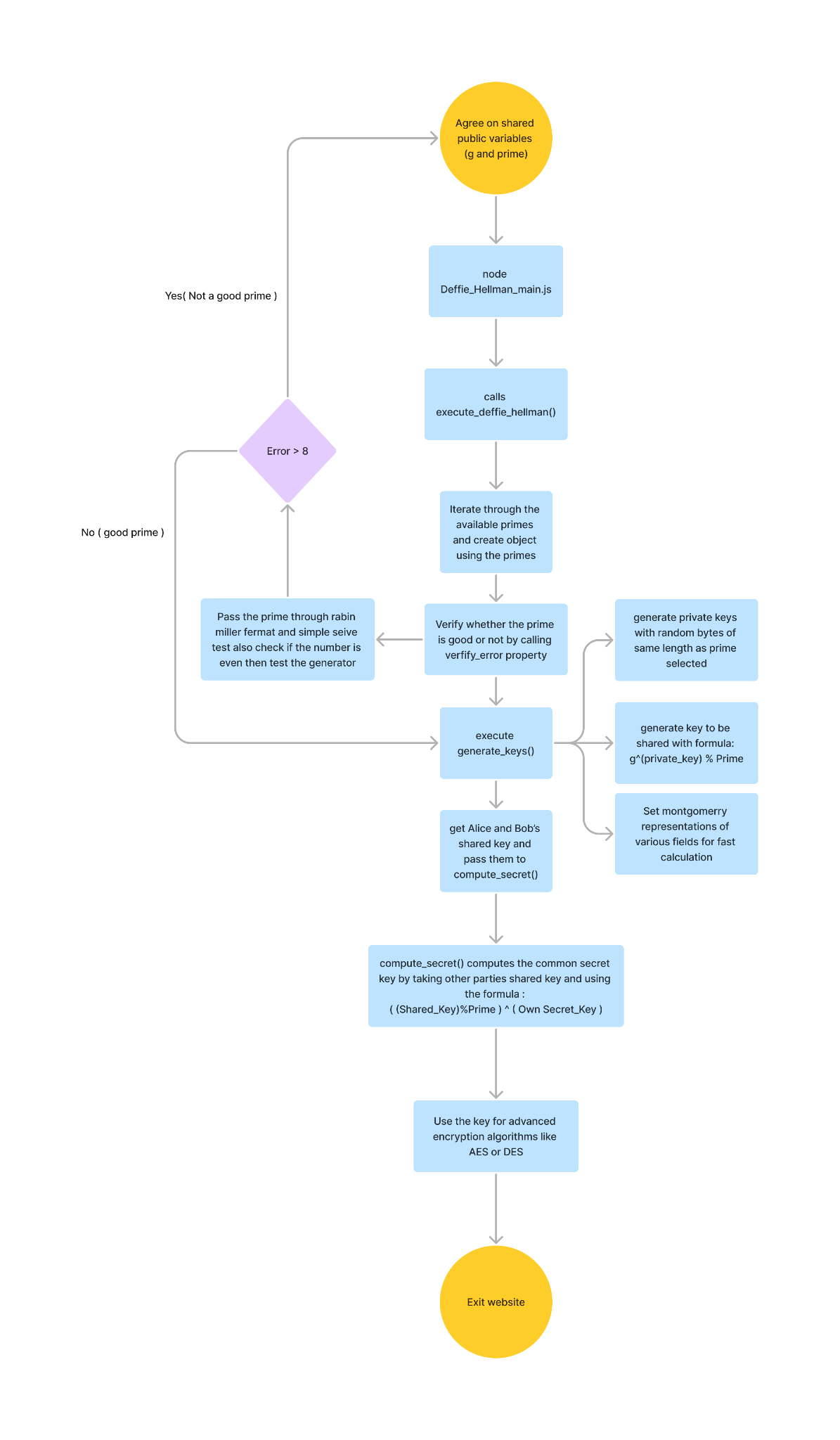
So there is a list of good primes . We have basically chosen primes from this list and made a primes.json file and stored them in it along with their g (i.e primitive root) as the primitive root for most primes are 2 or 5 and calculating them on the fly requires a lot of time so we just store them there. <https://www.ietf.org/rfc/rfc3526.txt>

Once we get the prime p and g we generate private keys (n) for our 2 parties in this case ‘Alice’ and ‘Bob’ which are just random bytes of same length as the prime chosen

| n & g | |
| --- | --- |
| Alice | Bob |
| a = Alice’s private key | b = Bob’s private key |
| K1A = | K1B = |
| K2A  = % n | K2B = % n |

Now as we can see K2A = K2B = shared secret key between 2 parties.

**Implementation :-**

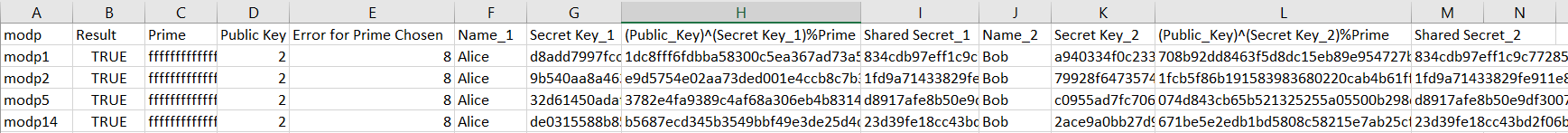


Libraries used

1. BN.js - Big Number javascript library is used to manage really big numbers like our primes in memory.
2. Papaparser.js - This library is used to generate the excel file and parse the results data for various tests into it for final
3. miller-rabin : The Miller-Rabin primality test is a probabilistic algorithm used to determine whether a given number is a probable prime or composite.
4. randombytes : Used for generating random bytes of number

**Result : -**

We show the output in a csv file where each row shows the results for various primes and public key pair(n,g) selected



Link to output file the screen shot is too big to fit in this [output](https://docs.google.com/spreadsheets/d/19j6yaGxuGnycuFX8qr-3J9zyCzzbhnlQ_GvO20jz_zs/edit?usp=sharing)